1983-09

# Implementation of a reliability shorthand on the Apple II+ microcomputer

Bartens, Eckhard

Monterey, California. Naval Postgraduate School

# NAVAL POSTGRADUATE SCHOOL
## Monterey, California

# THESIS

IMPLEMENTATION OF A RELIABILITY SHORTHAND
ON THE APPLE II+ MICROCOMPUTER

by

Eckhard Bartens

September 1983

Thesis Advisor: J. Esary

Approved for public release; distribution unlimited.

## REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

| | |
|---|---|
| 1. REPORT NUMBER | 2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER |
| 4. TITLE (and Subtitle): Implementation of a Reliability Shorthand on the Apple II+ Microcomputer | 5. TYPE OF REPORT & PERIOD COVERED: Master's Thesis; September 1983 |
| | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s): Eckhard Bartens | 8. CONTRACT OR GRANT NUMBER(s) |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93943 | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS |
| 11. CONTROLLING OFFICE NAME AND ADDRESS: Naval Postgraduate School, Monterey, California 93943 | 12. REPORT DATE: September 1983 |
| | 13. NUMBER OF PAGES: 55 |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) | 15. SECURITY CLASS. (of this report): Unclassified |
| | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

| | |
|---|---|
| Reliability Shorthand | Redundant System |
| Survival Function | Apple II+ |
| System Survival | Apple III |

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

It is shown how to use a reliability shorthand to get analytical results for small systems. The underlying probability distribution is the exponential function. An algorithm is described to convolve exponentially distributed random variables, and two computer programs are given to obtain numerical results for convolutions and mixed convolutions. After describing systems in shorthand notation, the programs can be used to

#20 - ABSTRACT - (CONTINUED)

compute reliabilities for those systems. The programs are coded in UCSD-Pascal for use with an Apple II+ and an Apple III.

Approved for public release; distribution unlimited.

# Implementation of a Reliability Shorthand on the Apple II+ Microcomputer

by

Eckhard Bartens
Captain, German Army
Dipl.-Ing., Fachhochschule des Heeres I, 1975

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL

September 1983

## ABSTRACT

It is shown how to use a reliability shorthand to get analytical results for small systems. The underlying probability distribution is the exponential function. An algorithm is described to convolve exponentially distributed random variables, and two computer programs are given to obtain numerical results for convolutions and mixed convolutions. After describing systems in shorthand notation, the programs can be used to compute reliabilities for those systems. The programs are coded in UCSD-Pascal for use with an Apple II+ and an Apple III.

TABLE OF CONTENTS

## LIST OF FIGURES

# I. INTRODUCTION

Every product eventually fails and, sooner or later, is no longer usable for its original purpose. The rate at which this deterioration process proceeds is called the failure rate. One is interested in predicting the time to failure for a component or an entire system. It is impossible to predict the time to failure precisely, but one can model this process, viewing the time to failure as a random variable which has a probability distribution that can be specified. There exist numerous life distributions which provide realistic probabilistic descriptions of the behavior of components or whole systems.

This paper is concerned only with the exponential distribution which plays a fundamental role in reliability. This distribution has many properties which facilitate the modelling process, and so encourage its application in many fields. Where not directly applicable, it is often used as a bound.

According to the exponential distribution, a component has a constant failure rate. An equivalent property is that the remaining life of the component is independent of its age: the "memoryless" or "no aging" or "no wear" property. This means that a used component is always as good as a new one.

Convolving life distributions for components to get the final life distribution for the system is rather difficult,

even for the exponential distribution. For an increasing number of components it's no longer possible to compute a closed form life distribution. This paper will show how to use a reliability shorthand notation to get analytical results for small basic systems, and later use an algorithm to compute numerical results with an Apple II+ microcomputer. The Pascal programs are provided in the appendices.

## II. RELIABILITY SHORTHAND

### A. NOTATION AND DEFINITIONS

The shorthand notation "EXP[$\lambda_1$] + EXP[$\lambda_2$]" denotes the distribution for a random variable $T_1+T_2$. $T_1$ has an exponential distribution with failure rate $\lambda_1$ and $T_2$ has an exponential distribution with failure rate $\lambda_2$. $T_1$ and $T_2$ are assumed to be independent.

In the following T always denotes a random life length and

$$\overline{F}(t) = \text{Probability } (T > t) = 1 - \overline{F}(t)$$

is the survival function as a function of time.

The convolution of life distributions corresponds to the summation of components with independent random lives to get the life distribution of a system.

## III. RELIABILITY FOR A ONE COMPONENT SYSTEM WITH SPARES

### A. GENERAL DESCRIPTION

There are two ways for a simple system, consisting of one active component and one spare, to survive a specified time t. Either the active component survives time t, in which case the spare need not be used, or the active component fails at some time s ($0 \leq s \leq t$) and the spare takes over and doesn't fail in the interval from s to t and thus completes the mission.

Let $\bar{F}_1(t)$ and $\bar{F}_2(t)$ be the exponential survival functions for component one and two respectively, and $f_1(t)$ and $f_2(t)$ their respective densities. Then the survival function $\bar{F}(t)$ for the system is denoted by:

$$\bar{F}(t) = \bar{F}_1(t) + \int_0^t \bar{F}_2(t-s) f_1(s) ds$$

where the integrand shows the probability that component one fails at time s ($f_1(s)$) and component two takes over for at least the length of time t-s. The following shows some examples how to get the survival function in closed form for different basic convolutions.

B. BASIC SURVIVAL FUNCTIONS

1. System with N Identical Components

The shorthand notation for this system is:

$$\text{EXP}[\lambda] + \text{EXP}[\lambda] + \ldots + \text{EXP}[\lambda]$$

Whenever an active component fails a spare takes over, as long as there is a spare. When the n-th component fails, the whole system stops functioning.

For n = 2:

$$\bar{F}(t) = \bar{F}_1(t) + \int_0^t \bar{F}_2(t-s) f_1(s) ds$$

$$= e^{-\lambda t} + \int_0^t e^{-\lambda(t-s)} \lambda e^{-\lambda s} ds$$

$$= (1 + \lambda t) e^{-\lambda t}, \qquad t \geq 0.$$

The density function for this survival function is:

$$f(t) = -\frac{d\bar{F}_1(t)}{dt} = -\frac{d[(1+\lambda t)e^{-\lambda t}]}{dt}$$

$$= -[(1+\lambda t)(-\lambda e^{-\lambda t}) + e^{-\lambda t}\lambda]$$

$$= -[-\lambda e^{-\lambda t} - \lambda^2 t e^{-\lambda t} + \lambda e^{-\lambda t}]$$

$$= \lambda^2 t e^{-\lambda t}$$

For n = 3:

The survival function for the two component system will be convolved with the survival function of the next component to obtain the survival function for the three component system.

$$\bar{F}(t) = (1+\lambda t)e^{-\lambda t} + \int_0^t e^{-\lambda(t-s)}\lambda^2 s e^{-\lambda s} ds$$

which leads to:

$$\bar{F}(t) = (1 + \lambda t + \frac{(\lambda t)^2}{2!})e^{-\lambda t}, \qquad t \geq 0.$$

For all n:

Continuing in this manner one gets the general solution for the survival function of this kind of system:

$$\bar{F}(t) = \frac{\sum_{i=1}^{n} (\lambda t)^{i-1}}{(i-1)e^{-\lambda t}}, \qquad t \geq 0.$$

This class of convolutions always gives the well-known Erlang[n,$\lambda$] survival function.

2. System with N Dissimilar Components

The shorthand notation for this system is denoted by:

$$EXP[\lambda_1] + EXP[\lambda_2] + \ldots + EXP[\lambda_n]$$

For n = 2:

The survival function for the active component is

$$\bar{F}_1(t) = e^{-\lambda_1 t}, \qquad t \geq 0.$$

The survival function for the spare is

$$\bar{F}_2(t) = e^{-\lambda_2 t}, \qquad t \geq 0.$$

The survival function for this system is

$$\bar{F}(t) = \bar{F}_1(t) + \int_0^t \bar{F}_2(t-s) f_1(s) ds$$

$$= e^{-\lambda_1 t} + \int_0^t e^{-\lambda_2 (t-s)} \lambda_1 e^{-\lambda_1 s} ds$$

$$= e^{-\lambda_1 t} + \frac{\lambda_1}{\lambda_1 - \lambda_2} e^{-\lambda_2 t} \int_0^t (\lambda_1 - \lambda_2) e^{-(\lambda_1 - \lambda_2) s} ds$$

$$= e^{-\lambda_1 t} + \frac{\lambda_1}{\lambda_1 - \lambda_2} e^{-\lambda_2 t} (1 - e^{-(\lambda_1 - \lambda_2) t})$$

$$= \frac{(\lambda_1 - \lambda_2) e^{-\lambda_1 t} + \lambda_1 e^{-\lambda_2 t} - \lambda_1 e^{-\lambda_1 t})}{(\lambda_1 - \lambda_2)}$$

$$= \frac{(\lambda_1 e^{-\lambda_2 t} - \lambda_2 e^{-\lambda_1 t})}{(\lambda_1 - \lambda_2)}, \qquad t \geq 0.$$

For n = 3:

The resulting survival function for the two component system will be convolved with the survival function of the next component to obtain the survival function for the three component system.

The result is

$$\bar{F}(t) = \frac{\lambda_2\lambda_3}{(\lambda_3-\lambda_1)(\lambda_2-\lambda_1)} e^{-\lambda_1 t} + \frac{\lambda_1\lambda_3}{(\lambda_3-\lambda_2)(\lambda_1-\lambda_2)} e^{-\lambda_2 t}$$

$$+ \frac{\lambda_1\lambda_2}{(\lambda_3-\lambda_2)(\lambda_3-\lambda_1)} e^{-\lambda_3 t}$$

For all n:

Continuing in this manner one gets the general solution for the survival function of this kind of system:

$$\bar{F}(t) = \sum_{i=1}^{n} \frac{\prod_{j\neq i} \lambda_j}{\prod_{j\neq i} (\lambda_j-\lambda_i)} e^{-\lambda_i t}$$

The two general survival functions for systems built from components which all have dissimilar failure rates or all have the same failure rate are very useful for computations of survival functions of systems which are formed from sub-systems of the above two kinds.

It will be shown in the following section how to use the above results to get a closed form for the survival function of a system which can not be computed with only one of those general formulas.

3. Systems With Identical and Dissimilar Components

Shorthand notation:

$$EXP[\lambda_1] + EXP[\lambda_1] + EXP[\lambda_2] + EXP[\lambda_2]$$

Let the random variables T be distributed in the following way:

$$T_1 \sim EXP[\lambda_1], \qquad T_2 \sim EXP[\lambda_1],$$

$$T_3 \sim EXP[\lambda_2], \qquad T_4 \sim EXP[\lambda_2].$$

There are different approaches available for computing the survival function for this system.

One can compute the survival function for the first two and then the second two components, using the general formula of Section III.B.1 and then convolve those two survival functions; or one can compute the survival function of the first two components and then add the other components one at a time. The latter approach will be used here to give a general idea of the procedure, and to lead to an algorithm for the numerical evaluation of the reliability of systems consisting of so many components that a closed form is no longer attainable or not attainable in an efficient manner.

The first two components ($T_1' = T_1$, $T_2' = T_1 + T_2$) have as shown in III.B.1 the survival function

$$\bar{F}_{T_2'}(t) = \bar{F}_{T_1'}(t) + \int_0^t \bar{F}_{T_2}(t-s) f_{T_1'}(s)\,ds$$

$$= (a_{11} + a_{12}t)e^{-\lambda_1 t}, \quad t \geq 0 \qquad (1)$$

with:

$$a_{11} = 1 \quad \text{and} \quad a_{12} = \lambda_1.$$

Adding the next component leads to $T_3' = T_1 + T_2 + T_3$ and the survival function

$$\bar{F}_{T_3'}(t) = \bar{F}_{T_2'}(t) + \int_0^t \bar{F}_{T_3'}(t-s) f_{T_2'}(s)\,ds$$

$$= (1 + \lambda_1 t)e^{-\lambda_1 t} + \int_0^t e^{-\lambda_2(t-s)} \lambda_1^2 s e^{-\lambda_1 s}\,ds$$

which leads after the integration to

$$\bar{F}_{T_3'}(t) = \frac{(\lambda_2^2 - 2\lambda_1\lambda_2) + (\lambda_1\lambda_2^2 - \lambda_1^2\lambda_2)t}{(\lambda_1 - \lambda_2)^2} e^{-\lambda_1 t} + \frac{\lambda_1^2}{(\lambda_1 - \lambda_2)^2} e^{-\lambda_2 t}$$

or in the form of equation (1) to

$$\bar{F}_{T_3'}(t) = (a_{11} + a_{12}t)e^{-\lambda_1 t} + a_{21}e^{-\lambda_2 t}$$

with

$$a_{11} = \frac{\lambda_2^2 - 2\lambda_1\lambda_2}{(\lambda_1 - \lambda_2)^2}, \quad a_{12} = \frac{\lambda_1\lambda_2^2 - \lambda_1^2\lambda_2}{(\lambda_1 - \lambda_2)^2}, \quad a_{21} = \frac{\lambda_1^2}{(\lambda_1 - \lambda_2)^2}.$$

Adding the last component leads to $T_4' = T_1 + T_2 + T_3 + T_4$ and the survival function:

$$\bar{F}_{T_4'}(t) = \bar{F}_{T_3'}(t) + \int_0^t \bar{F}_{T_4'}(t-s) f_{T_3'}(s) ds$$

$$= (a_{11}+a_{12}t)e^{-\lambda_1 t} + a_{21}e^{-\lambda_2 t}$$

$$+ \int_0^t e^{-\lambda_2(t-s)} [(a_{12}-a_{11}\lambda_1-a_{12}\lambda_1 s)e^{-\lambda_1 s}$$

$$- \lambda_2 a_{21} e^{-\lambda_2 s}] ds$$

Integrating and collecting terms gives the final survival function for the whole system:

$$\bar{F}_{T_4'}(t) = (a_{11}+a_{12}t)e^{-\lambda_1 t} + (a_{21}+a_{22}t)e^{-\lambda_2 t}, \quad t \geq 0$$

with

$$a_{11} = \frac{\lambda_2^3 - 3\lambda_1^2\lambda_2}{(\lambda_2 - \lambda_1)^3}, \qquad a_{12} = \frac{\lambda_1\lambda_2^2}{(\lambda_2-\lambda_1)^2},$$

$$a_{21} = \frac{\lambda_1^3 - 3\lambda_1^2\lambda_2}{(\lambda_2-\lambda_1)^3}, \qquad a_{22} = \frac{\lambda_1^2\lambda_2}{(\lambda_1-\lambda_2)^2}.$$

This section demonstrates that one can get the survival function by adding one random variable at a time and thus computing the new survival function. Each summation then gives a new or updated $a_{ij}$. Even though the integration is

not shown explicitly, one can imagine that this method of convolving survival functions is very tedious and it is very easy to make mistakes. So the next section introduces an algorithm to compute all the $a_{ij}$, which then makes it easy to evaluate the survival function at any time.

## C. ALGORITHM TO COMPUTE THE CONVOLUTION OF EXPONENTIAL LIVES

The shorthand notation to be solved is

$$\begin{aligned} & \text{EXP}[\lambda_1] + \text{EXP}[\lambda_1] + \ldots + \text{EXP}[\lambda_1] && (n_1 \text{ terms}) \\ + \ & \text{EXP}[\lambda_2] + \text{EXP}[\lambda_2] + \ldots + \text{EXP}[\lambda_2] && (n_2 \text{ terms}) \\ + \ & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots && \\ + \ & \text{EXP}[\lambda_k] + \text{EXP}[\lambda_k] + \ldots + \text{EXP}[\lambda_k] && (n_k \text{ terms}) \end{aligned}$$

The general solution for the survival function is of the form

$$\bar{F}(t) = A_1(t)e^{-\lambda_1 t} + A_2(t)e^{-\lambda_2 t} + \ldots + A_k(t)e^{-\lambda_k t}, \quad t \geq 0$$

with

$$\begin{aligned} A_1(t) &= a_{11} + a_{12}t + a_{13}t^2 + a_{14}t^3 + \ldots + a_{1n_1}t^{(n_1-1)}, \\ A_2(t) &= a_{21} + a_{22}t + a_{23}t^2 + a_{24}t^3 + \ldots + a_{2n_2}t^{(n_2-1)}, \\ &\vdots \\ A_k(t) &= a_{k1} + a_{k2}t + a_{k3}t^2 + a_{k4}t^3 + \ldots + a_{kn_k}t^{(n_k-1)}. \end{aligned}$$

An example shows how to get the survival function for the following shorthand notation:

$$\text{EXP}[\lambda_1] + \text{EXP}[\lambda_1] + \text{EXP}[\lambda_1]$$
$$+ \text{EXP}[\lambda_2] + \text{EXP}[\lambda_2]$$
$$+ \text{EXP}[\lambda_3]$$

Survival Function:

$$\bar{F}(t) = (a_{11}+a_{12}t+a_{13}t^2)e^{-\lambda_1 t} + (a_{21}+a_{22}t)e^{-\lambda_2 t} + a_{31}e^{-\lambda_3 t},$$

$$t \geq 0.$$

An algorithm to compute all the factors $a_{11}$, $a_{12}$, ..., $a_{31}$ and thus make it possible to convolve all the different exponential functions was earlier introduced by Gursel [Ref. 1], and is given here in a slightly modified form. This algorithm adds one exponential random variable in each run. The above example can be solved in this way by looping six times through this algorithm.

Notation for the algorithm:

$K$ = number of dissimilar failure rates (3 in above example);

$l_i$ = i-th type failure rate, i = 1,...,k;

$nn_i$ = number of random variables having the 9-th type failure rate;

$l_{ie}$ = failure rate of the ie-th lambda, ie = 1,...,K

$n_i$ = current number of random variables having the i-th failure rate, $n_i$ = 1,...,$nn_i$;

$a_{ij}$ = j-th coefficient of the i-th polynomial

ie = index for current polynomial and currently entering life.

Algorithm to compute the coefficients:

Initialization: $a_{jk} = 0$, for all j,k where $j = 1,\ldots,K$, $k = 1,\ldots,nn_i$

$n_i = 0$, for all i where $i = 1,\ldots,K$

Input: $\lambda_i$ for all i where $i = 1,\ldots,K$

$nn_i$ for all i where $i = 1,\ldots,K$.

After the first run: $a_{11} = 1$, $n_1 = 1$

$n_{ie} = n_{ie} + 1$, until $n_{ie} = nn_{ie}$

1. Update the coefficients $a_{ie,k}$ for $k = 2,\ldots,n_{ie}$

$$a_{ie,k} = \lambda_{ie} * a_{ie,m-1}/(m-1), \quad \text{where } m = n_{ie}-j \text{ for } j = 0,\ldots,n_{ie}-1$$

2. Update the coefficients $a_{ie,1}$

$$a_{ie,1} = a_{ie,1} + \sum_{\substack{i=1 \\ i\neq ie}}^{n_i} \left[\frac{a_{i1}\lambda_i}{(\lambda_i-\lambda_{ie})} + \sum_{\substack{j=2 \\ i\neq ie}}^{n_i} \frac{(j-1)!\lambda_{ie}a_{ij}}{(\lambda_i-\lambda_{ie})^j}\right]$$

3. Update the other coefficients $a_{ij}$ for all i,k where $i <> ie$, $n_i > 0$ for $i = 1,\ldots,K$, $k = 1,\ldots,n_i$

$$a_{i,n_i} = a_{i,n_i}\lambda_{ie}/(\lambda_{ie}-\lambda_i) \quad \text{for all } i \neq ie,$$

$$a_{i,n_i} = (a_{i,n_i}\lambda_{ie} - m a_{i,m})/(\lambda_{ie}-\lambda_i)$$ for all i where $i \neq ie$ and $n_i > 1$ for $m = n_i-j$, $j = 1,\ldots,(n_i-1)$

## D. PROGRAM FOR NUMERICAL EVALUATION OF THE RELIABILITY FOR A ONE COMPONENT SYSTEM WITH SPARES

The above algorithm is used to compute the reliability of a one component system with similar and/or dissimilar spares for any given time. The input to the program, which is written in PASCAL and listed in Appendix A, will be explained here with the same example as above:

$$\text{EXP}[\lambda_1] + \text{EXP}[\lambda_1] + \text{EXP}[\lambda_1] + \text{EXP}[\lambda_2] + \text{EXP}[\lambda_2] + \text{EXP}[\lambda_3]$$

Let's assume that $\lambda_1 = 0.1$, $\lambda_2 = 0.2$, $\lambda_3 = 0.3$ and we want to know the reliability for t = 0, 10, 20, ..., 90 hours. The prompts of the program are typed in large letters.

"HOW MANY DISSIMILAR LAMBDAS DO YOU WANT TO USE?"

3

"WRITE DOWN ALL THE DISSIMILAR LAMBDAS WITH AT LEAST ONE BLANK IN BETWEEN (INPUT NUMBERS < 1 WITH A LEADING ZERO) :"

0.1 0.2 0.3

"WRITE DOWN THE NUMBER OF EACH OF THE LAMBDAS WITH AT LEAST ONE BLANK IN BETWEEN:"

3 2 1

"HOW MANY TIME POINTS DO YOU WANT TO BE COMPUTED? (<=20) :"

10

"AT WHAT TIMES DO YOU WANT THE SURVIVAL FUNCTION TO BE EVALUATED? (TIME SHOULD HAVE THE SAME DIMENSION AS THE LAMBDAS AND SHOULD BE AT MOST 20 INTEGERS BETWEEN 0 AND 70.

THE UPPER LIMIT 70 IS REQUIRED ONLY, IF A GRAPHICAL PRESENTATION IS DESIRED LATER) :"

0 10 20 30 40 50 60 70 80 90

"DO YOU WANT A SIMULATION TO BE DONE PARALLEL TO THE COMPUTATIONS? (SERIOUS SIMULATION NEEDS A LOT OF TIME!) HIT 1 FOR "YES" AND "0" FOR NO:"

1

"HOW MANY NUMBERS DO YOU WANT TO USE AS BASIS FOR THE SIMULATION? :"

1000

After the computations the program will produce the following output:

```
FOLLOWING LAMBDAS WERE ENTERED:  0.1 0.2 0.3
            # OF EACH LAMBDA:    3   2   1

Results:
RELIABILITY ( T >  0 ) = 1.000000          (SIM = 1.000)
RELIABILITY ( T > 10 ) = 0.995799          (SIM = 0.996)
RELIABILITY ( T > 20 ) = 0.925259          (SIM = 0.925)
RELIABILITY ( T > 30 ) = 0.739626          (SIM = 0.728)
RELIABILITY ( T > 40 ) = 0.508910          (SIM = 0.489)
RELIABILITY ( T > 50 ) = 0.311271          (SIM = 0.334)
RELIABILITY ( T > 60 ) = 0.174439          (SIM = 0.173)
RELIABILITY ( T > 70 ) = 0.091597          (SIM = 0.068)
RELIABILITY ( T > 80 ) = 0.045784          (SIM = 0.036)
RELIABILITY ( T > 90 ) = 0.022028          (SIM = 0.020)
```

"DO YOU ALSO WANT A GRPAHICAL PRESENTATION OF THE DATA? (HIT "1" FOR "YES" AND "0" FOR "NO"):"

The graphical output will not be shown here, but will be generated from the program if desired.

Some notes on the program:

1. The program is set up to allow 20 dissimilar lambdas.
2. It is necessary to type leading zeros in PASCAL if one inputs decimal numbers.
3. The simulation generates n numbers (in this example n = 1000) for every random variable, to produce n observations on the whole system.
4. Because of
   - the behavior of the exponential function,
   - the sequence of the lambdas in the input,
   - the method of adding always only one random variable at a time,
   - the possible closeness of the values of the lambdas and
   - the possibly large number of the dissimilar lambdas,

   the program might have to compute with very large and small numbers and could produce erroneous results. Therefore a simulation is provided within the program to see if the computations are close to the simulation. Fixed limits which avoid errors can't be given, because all the possible errors are closely related. Fewer dissimilar lambdas for example would allow that their

values could be closer together, or that less care is needed with the sequence in the input, etc.

But after a long test phase and a comparison with results, produced with the IBM 3033, some hints can be given:

1. The input sequence of the lambdas should go from the smallest to the largest,
2. There should be no more than 20 lambdas altogether,
3. The difference in value of the dissimilar lambdas should be larger than 0.05.

## IV. RELIABILITY FOR A SYSTEM OF ACTIVE COMPONENTS AND SPARES

### A. MIXTURE OF CONVOLUTIONS

Before solving systems with active components and spares, the mixture of convolutions will be explained briefly. Let's assume that we want to compute the reliability of a parallel system with two different components, like that in Figure 1.

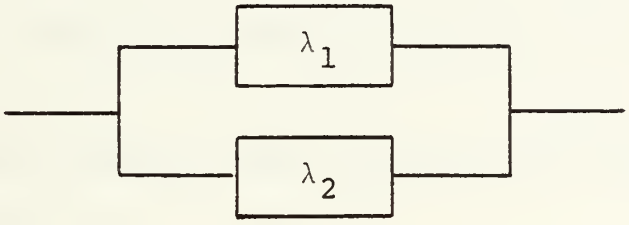


Figure 1. Simple Parallel System

The system starts with a life distributed as $EXP[\lambda_1+\lambda_2]$. Then there are two different possibilities: either component #1 fails with probability $P_1 = \lambda_1/(\lambda_1+\lambda_2)$ and the residual life is distributed as $EXP[\lambda_2]$. This is the upper branch of Figure 2. Or component #2 fails with probability $P_2 = \lambda_2/(\lambda_1+\lambda_2)$ and the residual life is distributed as $EXP[\lambda_1]$.

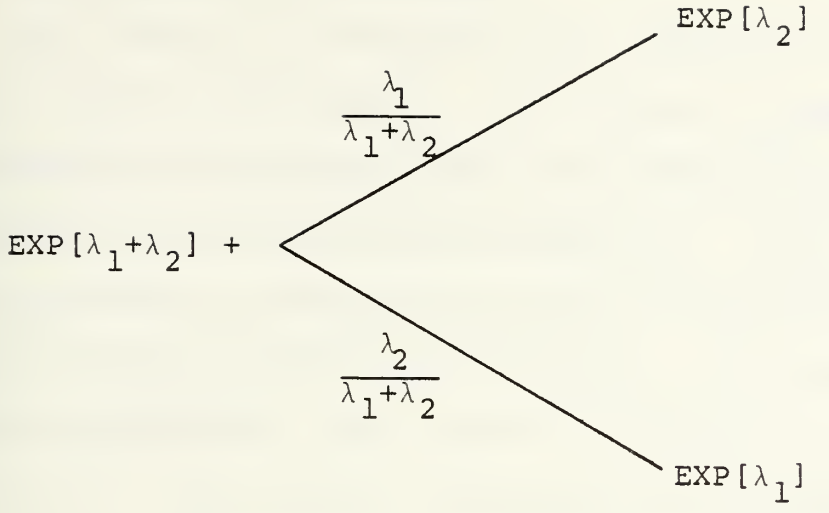


Figure 2. Shorthand Notation for Parallel System

Using the "MIX"-notation, the life can be written in the shorthand notation as:

$$EXP[\lambda_1+\lambda_2] + MIX\{P_1*EXP[\lambda_2],P_2*EXP[\lambda_1]\}.$$

Because of the distributive law, which is valid for these mixtures, one can transform this equation to

$$MIX\{P_1*EXP[\lambda_2] + EXP[\lambda_1+\lambda_2]),P_2*(EXP[\lambda_1] + EXP[\lambda_1+\lambda_2])\}$$

The first and the second expressions in parentheses multiplied by their respective probabilities, is a mixture of convolutions. The following section demonstrates how those mixtures can be solved numerically even if an analytical solution is not feasible.

B. DESCRIPTION OF THE SYSTEM

This section is concerned with computing the reliability of a system of components. The different components might be supported by spares, which have the same or a different failure rate as the original component. The following example will be used to illustrate the idea of describing the system in the reliability shorthand notation. The listing of a PASCAL program for these types of systems is provided in Appendix B. The program incorporates the method of program #1 to add the components, one at a time, to get first the convolution of the exponential distributions and then via the mixture the reliability of the whole system.

Example:

The system consists of two parallel components (#1 and #2) in series with a third component (#3). A spare is provided for component #1. The components have the failure rate $\lambda_1$, $\lambda_2$ and $\lambda_3$ respectively. The spare has the same failure rate as the component it can replace.

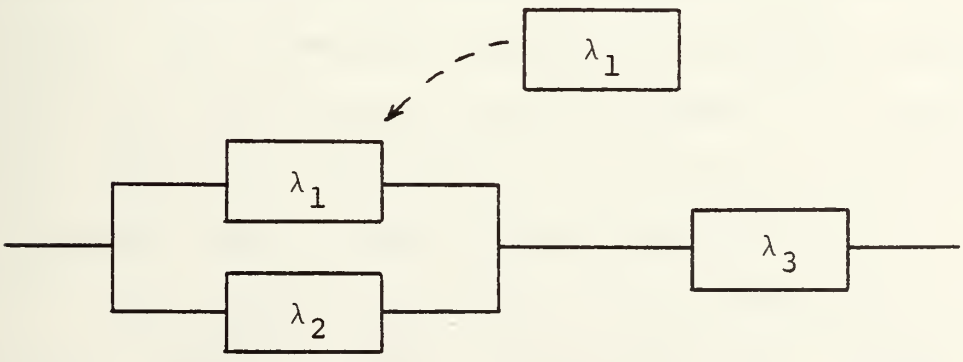


Figure 3. Combined Serial/Parallel System with Spare

C. SHORTHAND BRANCHING DIAGRAM

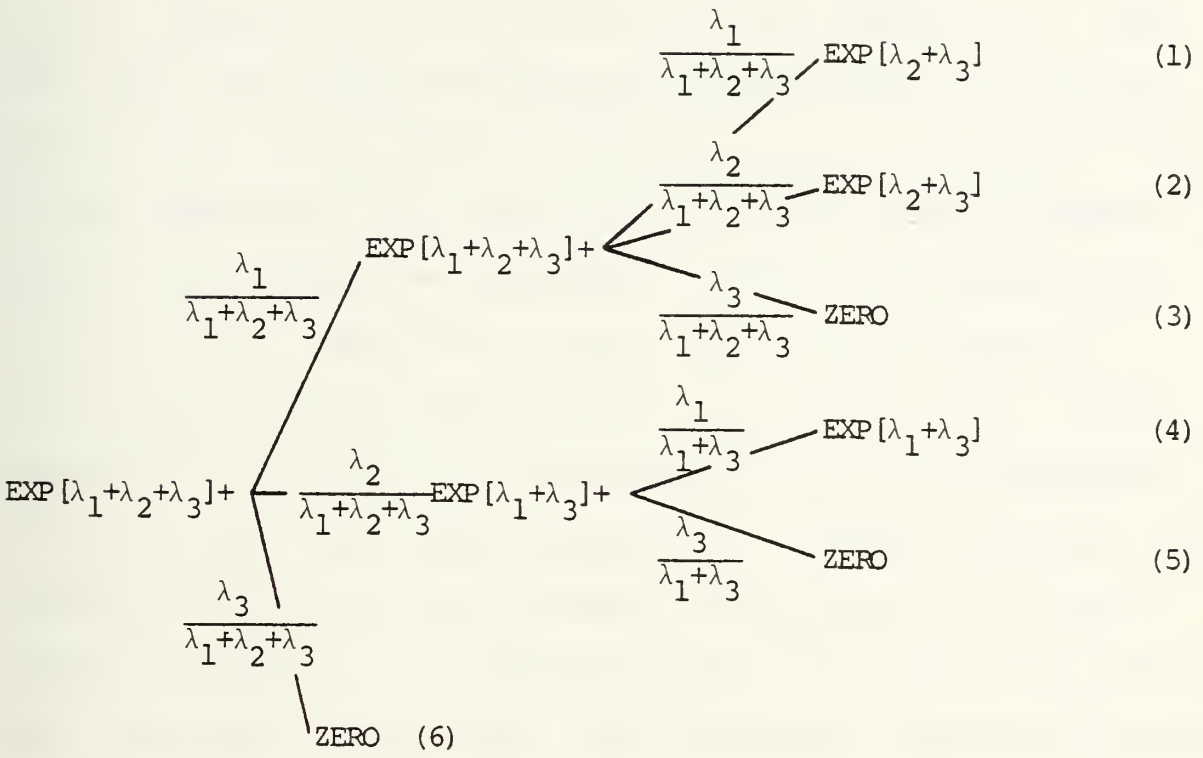


Figure 4. Shorthand Branching Diagram for Serial/ Parallel System with Spare

The numbers in the numbered parentheses in the figure above describe the different endpoints (leaves) in this notation. The connections between the original system (root) and the leaves will be called branches. In the following: the connection from the root to leaf #1 is called branch #1, from the root to leaf #2 branch #2, etc.

The original system has a life distributed as EXP[$\lambda_1+\lambda_2+\lambda_3$]. Three different things could happen now. Either component #1 fails with probability $\lambda_1/(\lambda_1+\lambda_2+\lambda_3)$ or component #2 fails

with probability $\lambda_2/(\lambda_1+\lambda_2+\lambda_3)$ or component #3 fails with probability $\lambda_3/(\lambda_1+\lambda_2+\lambda_3)$. Let's continue in branch #1 and assume component #1 failed with the above probability. Now the spare can replace the failed component and the system's life is again distributed as $EXP[\lambda_1+\lambda_2+\lambda_3]$. The system is now in the same state as at the beginning and the same arguments hold. Let's assume new component #1 fails with probability $\lambda_1/(\lambda_1+\lambda_2+\lambda_3)$, then the remaining life is distributed as $EXP[\lambda_2+\lambda_3]$. If component #2 fails (the probability of this failure is $\lambda_2(\lambda_1+\lambda_2+\lambda_3)$), the remaining life is distributed as $EXP[\lambda_1+\lambda_3]$. The third possibility, component #3 fails with probability $\lambda_3/(\lambda_1+\lambda_2+\lambda_3)$, brings the failure of the whole system, denoted by the "ZERO"-notation. This says that there is a remaining life of zero. The reason why this third branch isn't simply omitted is that even if there is no remaining life, there is a specified probability that this event occurs. And this has to be taken into account to compute the total life of the system.

The other three branches of the above figure are developed analogously.

## D. THE SURVIVAL FUNCTION

Let's denote the survival function of branch #1 by $\bar{F}_1(t)$, and the probability that the events in branch #1 occur in this special sequence as $P_1$. Then the life of branch #1 is distributed as $P_1*(EXP[\lambda_1+\lambda_2+\lambda_3] + EXP[\lambda_1+\lambda_2+\lambda_3] + EXP[\lambda_2+\lambda_3])$

and the survival function of this system is the following mixture:

$$\bar{F}(t) = MIX\{p_1*\bar{F}_1(t), p_2*\bar{F}_2(t), p_3*\bar{F}_3(t), p_4*\bar{F}_4(t), p_5*\bar{F}_5(t), p_6*\bar{F}_6(t)\}$$

For the computation of the coefficients for every single survival function the algorithm of Section III.C can be used again.

## E. PROGRAM FOR THE NUMERICAL EVALUATION OF THE RELIABILITY FOR SYSTEMS WITH SEVERAL ACTIVE COMPONENTS AND SPARES

This section describes the program for computing the reliability for systems as described above. While reading this part, one should look at Figure 4. The program is interactive. Only the input of Figure 4 will be explained here.

| Prompt | Input |
|---|---|
| "How many branches?" | 6 CR (Carriage return) |
| 1. Branch : | |
| 1. Element : | 2 CR 3 CR 0 CR |
| 2. Element : | 1 CR 1 CR 2 CR 3 CR 0 CR |
| 3. Element : | 1 CR 2 CR 3 CR 0 CR |
| 4. Element : | 1 CR 1 CR 2 CR 3 CR 0 CR |
| 5. Element : | 1 CR 2 CR 3 CR 0 CR |
| 6. Element : | 0 CR |
| 2. Branch : | |
| 1. Element : | 1 CR 3 CR 0 CR |
| 2. Element : | 2 CR 1 CR 2 CR 3 CR 0 CR |
| 3. Element : | 1 CR 2 CR 3 CR 0 CR |

4. Element : 1 CR 1 CR 2 CR 3 CR 0 CR
5. Element : 1 CR 2 CR 3 CR 0 CR
6. Element : 0 CR

3. Branch :
1. Element : 0 CR
2. Element : 3 CR 1 CR 2 CR 3 CR 0 CR
3. Element : 1 CR 2 CR 3 CR 0 CR
4. Element : 1 CR 1 CR 2 CR 3 CR 0 CR
5. Element : 1 CR 2 CR 3 CR 0 CR
6. Element : 0 CR

4. Branch :
1. Element : 1 CR 3 CR 0 CR
2. Element : 1 CR 1 CR 3 CR 0 CR
3. Element : 1 CR 3 CR 0 CR
4. Element : 2 CR 1 CR 2 CR 3 CR 0 CR
5. Element : 1 CR 2 CR 3 CR 0 CR
6. Element : 0 CR

5. Branch :
1. Element : 0 CR
2. Element : 3 CR 1 CR 3 CR 0 CR
3. Element : 1 CR 3 CR 0 CR
4. Element : 2 CR 1 CR 2 CR 3 CR 0 CR
5. Element : 1 CR 2 CR 3 CR 0 CR
6. Element : 0 CR

6. Branch :
1. Element : 0 CR
2. Element : 3 CR 1 CR 2 CR 3 CR 0 CR
3. Element : 1 CR 2 CR 3 CR 0 CR
4. Element : 0 CR

The branches in the figure count from top to bottom and the elements in each branch from the leaf to the root. The first element in the first branch is EXP[$\lambda_2+\lambda_3$] and one inputs only the two indices of the lambdas. The second element in the same branch is $\lambda_1/(\lambda_1+\lambda_2+\lambda_3)$ and one inputs first the index of the lambda in the numerator and then the indices of the denominator. After each element input a "0". If there is no sixth element as in branch #1 to branch #5, no fourth element as in branch #6, or no element as the ZERO elements in branches #3, #5, and #6, simply input a "0". Because of the data structure used in the PASCAL program, the system always has to be split up into pieces, so that there is only a single failure in each step. Because of the algorithm used, there shouldn't be more than 20 elements in each branch to avoid errors.

## V. SUMMARY

The reliability shorthand is a convenient way to describe the life of different systems. This shorthand then is the underlying data structure to compute numerically the reliability for systems for which it is difficult or impossible to get the survival function in closed form.

```
PROGRAM CONVOLUTION;
USES TRANSCEND,APPLESTUFF;
CONST
  ASTERISK='*';
  BLANK   =' ';
  DASH    ='-';
  LINE    ='!';
TYPE
  ARR = PACKED ARRAY[1..20] OF REAL;
  ARI = PACKED ARRAY[1..20] OF INTEGER;
VAR
  A:PACKED ARRAY[1..20,1..20] OF REAL;
  L,PRO,REL,RELSIM :ARR;
  NINIT,NI,T:ARI;
  KK,JJ,IE,K,TMAX,SIMMAX,Y,X,COUNT,VAL:INTEGER;
  I,II,J,FACT,NN,NKK,NNI,CHGR,CHSIM:INTEGER;
  IT,D :INTEGER;  (* INDEX FOR T-ARRAY *)
  SUM,RELI:REAL;
  PRINTVAR:CHAR;
  (* L = ARRAY OF DISSIMILAR FAILURE RATES *)
  (* NI[I] = NUMBER OF LIVES HAVING THE ITH FAILURE RATE *)
  (* A[I,J] = COEFFICIENT FOR THE POLYNOMIAL *)
  (* T = ARRAY OF TIMES AT WHICH THE RELIABILITY WILL BE
  COMPUTED *)

PROCEDURE ATTENTION;
VAR
  DURATION,I:INTEGER;
  PITCH:ARRAY[1..10] OF INTEGER;
BEGIN
  PITCH[1]:= 15;
  PITCH[2]:= 17;
  PITCH[3]:= 19;
  PITCH[4]:= 20;
  PITCH[5]:= 22;
  DURATION:= 30;
  FOR I:=1 TO 5 DO
    NOTE(PITCH[I],DURATION);
END;

PROCEDURE DECIDE2;
BEGIN
  WRITELN;WRITELN;
  WRITELN('DO YOU WANT TO CONTINUE WITH THE SAME PROGRAM?
  TYPE "U" :');
  WRITELN('DO YOU WANT TO COMPUTE MIXED CONVOLUTIONS?
  TYPE "XP2" :');
  EXIT(P1);
```

```
END;

PROCEDURE INPUT;
BEGIN
  WRITELN;WRITELN;
  GOTOXY(0,12);
  WRITELN('HOW MANY DISSIMILAR LAMBDAS DO YOU WANT TO USE?  ');
  READLN(K);
  WRITELN;
  WRITELN('WRITE DOWN ALL THE DISSIMILAR LAMBDAS WITH AT
  LEAST ONE BLANK IN BETWEEN:');
  WRITELN('(NUMBERS < 1 NEED A LEADING ZERO!)');
  FOR I:=1 TO K DO
    READ(L[I]);
  WRITELN;
  WRITE('WRITE DOWN THE NUMBER OF EACH OF THE LAMBDAS WITH AT
  LEAST ONE BLANK IN');
  WRITELN('BETWEEN:');
  FOR I:=1 TO K DO
    READ(NINIT[I]);
  WRITELN;
  WRITE('HOW MANY TIME POINTS DO YOU WANT TO BE COMPUTED?
  ( <= 20 ) : ');
  READLN(TMAX);
  WRITELN('AT WHAT TIME DO YOU WANT THE SURVIVAL FUNCTION TO
  BE EVALUATED?');
  WRITELN('(INTEGER TIMES BETWEEN 0-60 IN THE SAME UNITS AS
  THE LAMBDAS)');
  FOR I:=1 TO TMAX DO
    READ(T[I]);
  WRITELN;
  WRITELN('DO YOU WANT A SIMULATION TO BE DONE PARALLEL TO
  THE COMPUTATIONS?');
  WRITELN('(SERIOUS SIMULATION NEEDS A LOT OF TIME!)');
  WRITE('HIT "1" FOR YES OR "0" FOR NO :  ');
  READLN(CHSIM);
  WRITELN;
  IF CHSIM = 1 THEN
    BEGIN
      WRITELN('HOW MANY NUMBERS DO YOU WANT TO USE AS BASIS
      FOR THE SIMULATION?');
      READLN(SIMMAX);
    END;
  WRITELN;WRITELN;
END;

PROCEDURE OUTPUT;
BEGIN
  WRITE('FOLLOWING ',K:3,' LAMBDAS WERE ENTERED :');
  FOR I:=1 TO K DO
    WRITE(L[I]:5:2);
  WRITELN;
```

```
  WRITE('                    # OF EACH LAMBDA :');
  FOR I:=1 TO K DO
    WRITE(NINIT[I]:5);
  WRITELN;WRITELN;WRITELN;
end;

PROCEDURE INITIALIZE;
BEGIN
  KK:=1;
  WHILE KK<=K DO
    BEGIN
      A[KK,1]:=1.0;
      NI[KK]:=1;
      JJ:=1;
      WHILE JJ<=K DO
        BEGIN
          IF JJ<>KK THEN
            BEGIN
              A[KK,1]:=A[KK,1]*L[JJ]/(L[JJ]-L[KK]);
            END;
          JJ:=JJ+1;
        END;
      KK:=KK+1;
    END;
    KK:=K;
    IE:=1;
END;

PROCEDURE SUMMATION;
BEGIN
  SUM:=0.0;
  FOR I:=1 TO K DO
    BEGIN
      IF I<>IE THEN
        BEGIN
          SUM:=SUM+A[I,1]*L[I]/(L[I]-L[IE]);
          IF NI[I]>=2 THEN
            BEGIN
              FACT:=1;
              NKK:=NI[I];
              FOR II:=2 TO NKK DO
                BEGIN
                  FACT:=FACT*(II-1);
                  IF L[I]>L[IE] THEN
                    SUM:=SUM+A[I,II]*FACT*L[IE]/EXP(II*LN(
                    L[I]-L[IE]))
                  ELSE
                    BEGIN
                      IF (II MOD 2)=0 THEN
                        SUM:=SUM+A[I,II]*FACT*L[IE]/EXP(
                        II*LN(ABS(L[I]-L[IE])))
                      ELSE
```

```
                        SUM:=SUM+A[I,II]*FACT*L[IE]/(-EXP(
                        II*LN(ABS(L[I]-L[IE]))))
                    END;
                END;
            END;
        END;
    END;
    A[IE,1]:=SUM+A[IE,1];
END;

PROCEDURE SUB1;
BEGIN
  I:=1;
  WHILE I<=K DO
    BEGIN
      IF I<>IE THEN
        BEGIN
          A[I,NI[I]]:=A[I,NI[I]]*L[IE]/(L[IE]-L[I]);
          IF NI[I]<>1 THEN
            BEGIN
              J:=1;
              NNI:=NI[I]-J;
              WHILE J<=(NI[I]-1) DO
                BEGIN
                  A[I,NNI]:=(L[IE]*A[I,NNI]-NNI*A[I,NNI+1])/
                  (L[IE]-L[I]);
                  J:=J+1;
                  NNI:=NI[I]-J;
                END;
            END;
        END;
      I:=I+1;
    END;
END;

PROCEDURE ONEATATIME;
BEGIN
  IF NINIT[IE]=NI[IE] THEN
    IE:=IE+1
  ELSE
    BEGIN
      NI[IE]:=NI[IE]+1;
      J:=0;
      REPEAT
        NN:=NI[IE]-J;
        A[IE,NN]:=L[IE]*A[IE,NN-1]/(NN-1);
        J:=J+1;
      UNTIL NN=2;
      SUMMATION;
      SUB1;
    END;
END;
```

```
PROCEDURE RAND;
VAR
  I,II,III,COUNT,J:INTEGER;
  R,TSIM:REAL;
BEGIN
  III:=0;
  COUNT:=0;
  REPEAT
    TSIM:=0.0;
    II:=1;
    FOR J:=1 TO K DO
      BEGIN
        FOR I:=1 TO NINIT[J] DO
          BEGIN
            R:=RANDOM/MAXINT;
            TSIM:=TSIM+(-(LN(R)/L[III]));
          END;
        II:=II+1;
      END;
    IF TSIM>T[IT] THEN
      COUNT:=COUNT+1;
    III:=III+1;
  UNTIL III=SIMMAX;
  RELSIM[IT]:=COUNT/SIMMAX;
END;

PROCEDURE SUB2;
BEGIN
  RELI:=0.0;
  IF T[IT]=0 THEN
    RELI:=1.0
  ELSE
    BEGIN
      FOR I:=1 TO K DO
        BEGIN
          SUM:=0.0;
          NKK:=NI[I];
          FOR J:=1 TO NKK DO
            SUM:=SUM+A[I,J]*EXP((J-1)*LN(T[IT]));
          PRO[I]:=SUM*EXP(-L[I]*T[IT]);
        END;
        FOR I:=1 TO K DO
          RELI:=RELI+PRO[I];
    END;
    IF RELI>1.0 THEN
      REL[IT]:=1.0
    ELSE
      REL[IT]:=RELI;
    IF CHSIM=1 THEN
      RAND;
END;
```

```
PROCEDURE YAXIS;
BEGIN
  IF (X=0) THEN PRINTVAR:=LINE;
  IF ((X=0) AND (Y=20)) AND (T[1]=0) THEN PRINTVAR:=ASTERISK;
  IF (X=-3) AND (Y MOD 2 = 0) THEN PRINTVAR:='0';
  IF (X=-3) AND (Y=20) THEN PRINTVAR:='1';
  IF (X=-2) AND (Y MOD 2=0) THEN PRINTVAR:='.';
  IF (X=-1) AND ((Y=20) OR (Y=0)) THEN PRINTVAR:='0';
  IF (X=-1) AND (Y=18) THEN PRINTVAR:='9';
  IF (X=-1) AND (Y=16) THEN PRINTVAR:='8';
  IF (X=-1) AND (Y=14) THEN PRINTVAR:='7';
  IF (X=-1) AND (Y=12) THEN PRINTVAR:='6';
  IF (X=-1) AND (Y=10) THEN PRINTVAR:='5';
  IF (X=-1) AND (Y=8) THEN PRINTVAR:='4';
  IF (X=-1) AND (Y=6) THEN PRINTVAR:='3';
  IF (X=-1) AND (Y=4) THEN PRINTVAR:='2';
  IF (X=-1) AND (Y=2) THEN PRINTVAR:='1';
END;

PROCEDURE GRAPHICS;
BEGIN
  COUNT:=1;
  FOR Y:=20 DOWNTO 1 DO
    BEGIN
      FOR X:=-3 TO 76 DO
        BEGIN
          IF COUNT<=TMAX THEN
            BEGIN
              VAL:=ROUND(20*REL[COUNT]);
              IF (VAL=Y) AND (X=T[COUNT]) THEN
                BEGIN
                  PRINTVAR:=ASTERISK;
                  COUNT:=COUNT+1;
                END
              ELSE
                PRINTVAR:=BLANK;
            END
          ELSE
            PRINTVAR:=BLANK;
          YAXIS;
          WRITE(PRINTVAR);
        END;
  END;
  Y:=0;
  FOR X:=-3 TO 76 DO
    BEGIN
      IF (X MOD 5 =0) THEN PRINTVAR:=LINE;
      IF COUNT<=TMAX THEN
        BEGIN
          VAL:=ROUND(20*REL[COUNT]);
          IF (VAL=Y) AND (X=T[COUNT]) THEN
```

```
            BEGIN
              PRINTVAR:=ASTERISK;
              COUNT:=COUNT+1;
            END
          ELSE
            IF (X MOD 5 <> 0) THEN
              PRINTVAR:=DASH;
        END
      ELSE
        IF (X MOD 5 <> 0) THEN
          PRINTVAR:=DASH;
      IF (X=-3) THEN PRINTVAR:='0';
      IF (X=-2) THEN PRINTVAR:='.';
      IF (X=-1) THEN PRINTVAR:='0';
      WRITE(PRINTVAR);
    END;
  WRITE('    0    5    10    15    20    25    30    35    40    45
  50    55    60');
  WRITELN('     TIME -->');
END;


BEGIN
  INPUT;
  OUTPUT;
  FOR IT:=1 TO TMAX DO
    BEGIN
      INITIALIZE;
      REPEAT;
        ONEATATIME;
      UNTIL IE=K+1;
      SUB2;
      IF CHSIM=1 THEN
        WRITELN('RELIABILITY ( T >',T[IT]:3,' ) = ',REL[IT]:8:6,
        '     (SIM =',RELSIM[IT]:5:3,')')
      ELSE
        WRITELN('RELIABILITY ( T >',T[IT]:3,' ) = ',REL[IT]:8:6);
    END;
  WRITELN;WRITELN;
  ATTENTION;
  WRITELN('DO YOU ALSO WANT A GRAPHICAL PRESENTATION OF
  THE DATA?');
  WRITE('HIT "1" FOR YES OR "0" FOR NO :  ');
  READLN(CHGR);
  WRITELN;
  WRITELN;
  IF CHGR=1 THEN
    GRAPHICS;
  DECIDE2;
END.
```

# APPENDIX B

```
PROGRAM MIXED_CONVOLUTION;
USES TRANSCEND,APPLESTUFF;
CONST
  ASTERISK='*';
  BLANK   =' ';
  DASH    ='-';
  LINE    ='!';
  MAX     =10;
TYPE
  ARR = PACKED ARRAY[1..MAX] OF REAL;
  ARI = PACKED ARRAY[1..MAX] OF INTEGER;
  DARR = PACKED ARRAY[1..MAX,1..MAX] OF REAL;
VAR
  AR:PACKED ARRAY[0..MAX,0..MAX,0..MAX] OF INTEGER;
  A:PACKED ARRAY[1..MAX,1..MAX] OF REAL;
  SUMREL:PACKED ARRAY[0..MAX] OF REAL;
  N:PACKED ARRAY[0..MAX] OF INTEGER;
  NINIT,NI,T,ZA,INP:ARI;
  PROB,RELTOT,LAMB:DARR;
  L,PRO,REL,P:ARR;
  I,J,K,BR,BRANCHES,COUNT,KK,IL,KKK,JJ,IE:INTEGER;
  IT,II,FA,NN,NKK,NNI,CHGR,TMAX,Y,X,VAL:INTEGER;
  SUM,RELI,LAM,DEN:REAL;
  PRINTVAR,CH:CHAR;
  (* L = ARRAY OF ALL LAMBDA *)
  (* PRO[I] = PROBABILITY OF EACH BRANCH *)
  (* T = ARRAY OF ALL TIMES AT WHICH THE RELIABILITY WILL
  BE COMPUTED *)
  (* BRANCHES = NUMBER OF BRANCHES IN SYSTEM *)

PROCEDURE INPUT;
BEGIN
  WRITELN('HOW MANY DISSIMILAR LAMBDAS ARE IN THE SYSTEM?  ');
  READLN(K);
  WRITELN('WRITE DOWN ALL THE LAMBDAS WITH AT LEAST ONE BLANK
  IN BETWEEN:');
  WRITELN('(NUMBERS < 1 NEED A LEADING ZERO!)');
  FOR I:=1 TO K DO
    READ(L[I]);
  WRITELN('HOW MANY TIME POINTS DO YOU WANT TO BE COMPUTED?
  ( <= 20 ) : ');
  READLN(TMAX);
  WRITELN('AT WHAT TIME DO YOU WANT THE SURVIVAL FUNCTION TO
  BE EVALUATED?');
  WRITELN('(INTEGER TIMES BETWEEN 0-60 IN THE SAME UNITS AS
  THE LAMBDAS)');
  FOR I:=1 TO TMAX DO
    READ(T[I]);
```

```
  WRITELN;WRITELN;
END;

PROCEDURE ATTENTION;
VAR
  DURATION,I:INTEGER;
  PITCH:ARRAY[1..10] OF INTEGER;
BEGIN
  PITCH[1]:= 15;
  PITCH[2]:= 17;
  PITCH[3]:= 19;
  PITCH[4]:= 20;
  PITCH[5]:= 22;
  DURATION:= 30;
  FOR I:=1 TO 5 DO
    NOTE(PITCH[I],DURATION);
END;

PROCEDURE OUTPUT1;
BEGIN
  WRITE('FOLLOWING ',K:3,' LAMBDAS WERE ENTERED :');
  FOR I:=1 TO K DO
    WRITE(L[I]:5:2);
  WRITELN;
END;

PROCEDURE OUTPUT2;
BEGIN
  FOR I:=1 TO BRANCHES DO
    BEGIN
      FOR J:=1 TO MAX DO
        BEGIN
          FOR K:=1 TO MAX DO
            BEGIN
              IF (AR[I,J,K]=0) AND (K<>1) THEN
                AR[I,J,K]:=-1;
            END;
        END;
    END;
END;

PROCEDURE GETDATA;
BEGIN
  FOR I:=0 TO MAX DO
    FOR J:=0 TO MAX DO
      FOR K:=0 TO MAX DO
        AR[I,J,K]:=-1;
  N[0]:=1;
  WRITELN;
  WRITELN('HOW MANY BRANCHES?');
  READLN(BRANCHES);
  I:=0;
```

```
  REPEAT
    I:=I+1;
    WRITELN(I:3,'. BRANCH: ');
    J:=0;
    REPEAT
      J:=J+1;
      WRITELN(J:3,'. ELEMENT: ');
      K:=0;
      REPEAT
        K:=K+1;
        READ(ZA[K]);
        AR[I,J,K]:=ZA[K];
        N[J]:=K;
      UNTIL AR[I,J,K]=0;
    UNTIL ((AR[I,J-1,N[J-1]]=0) AND (AR[I,J,N[J]]=0))
    AND (ZA[1]=0);
  UNTIL I=BRANCHES;
  OUTPUT2;
END;

PROCEDURE COMPPROB;
BEGIN
  FOR I:=1 TO MAX DO
    FOR J:=1 TO MAX DO
      PROB[I,J]:=-1;
  I:=1;
  J:=2;
  WHILE I<=BRANCHES DO
    BEGIN
      DEN:=0;
      K:=2;
      COUNT:=1;
      WHILE AR[I,J,K]>0 DO
        BEGIN
          WHILE AR[I,J,K]>0 DO
            BEGIN
              DEN:=DEN+L[AR[I,J,K]];
              K:=K+1;
            END;
          PROB[I,COUNT]:=L[AR[I,J,1]]/DEN;
          COUNT:=COUNT+1;
          J:=J+2;
          K:=2;
          DEN:=0;
        END;
        I:=I+1;
        J:=2;
    END;
  FOR I:=1 TO BRANCHES DO
    BEGIN
      J:=1;
      P[I]:=1;
```

```
      WHILE PROB[I,J]>0 DO
        BEGIN
          P[I]:=P[I]*PROB[I,J];
          J:=J+1;
        END;
    END;
  WRITELN;
END;

PROCEDURE COMPLAMB;
BEGIN
  FOR I:=1 TO MAX DO
    FOR J:=1 TO MAX DO
      LAMB[I,J]:=-1;
  I:=1;
  J:=1;
  WHILE I<=BRANCHES DO
    BEGIN
      LAM:=0;
      K:=1;
      COUNT:=1;
      WHILE AR[I,J,K]>=0 DO
        BEGIN
          WHILE AR[I,J,K]>0 DO
            BEGIN
              LAM:=LAM+L[AR[I,J,K]];
              K:=K+1;
            END;
          LAMB[I,COUNT]:=LAM;
          COUNT:=COUNT+1;
          J:=J+2;
          K:=1;
          LAM:=0;
        END;
        I:=I+1;
        J:=1;
    END;
END;

PROCEDURE SORT;
VAR
  TEMP:REAL;
  SORTED:BOOLEAN;
BEGIN
  REPEAT
    I:=1;
    SORTED:=TRUE;
    WHILE L[I+1]>0 DO
      BEGIN
        IF L[I]>L[I+1] THEN
          BEGIN
            TEMP:=L[I];
```

```
            L[I]:=L[I+1];
            L[I+1]:=TEMP;
            SORTED:=FALSE;
          END;
        I:=I+1;
      END;
  UNTIL SORTED=TRUE;
  L[I+1]:=-1;
  L[I+2]:=-1;
END;

PROCEDURE INITIALIZE;
BEGIN
  KK:=1;
  WHILE KK<=K DO
    BEGIN
      A[KK,1]:=1.0;
      NI[KK]:=1;
      JJ:=1;
      WHILE JJ<=K DO
        BEGIN
          IF JJ<>KK THEN
            BEGIN
              A[KK,1]:=A[KK,1]*L[JJ]/(L[JJ]-L[KK]);
            END;
          JJ:=JJ+1;
        END;
      KK:=KK+1;
    END;
    KK:=K;
    IE:=1;
END;

PROCEDURE SUMMATION;
BEGIN
  SUM:=0.0;
  FOR I:=1 TO K DO
    BEGIN
      IF I<>IE THEN
        BEGIN
          SUM:=SUM+A[I,1]*L[I]/(L[I]-L[IE]);
          IF NI[I]>=2 THEN
            BEGIN
              FA:=1;
              NKK:=NI[I];
              FOR II:=2 TO NKK DO
                BEGIN
                  FA:=FA*(II-1);
                  IF L[I]>L[IE] THEN
                    SUM:=SUM+A[I,II]*FA*L[IE]/EXP(II*LN(
                    L[I]-L[IE]))
                  ELSE
```

```
                    BEGIN
                      IF (II MOD 2)=0 THEN
                        SUM:=SUM+A[I,II]*FA*L[IE]/EXP(
                        II*LN(ABS(L[I]-L[IE])))
                      ELSE
                        SUM:=SUM+A[I,II]*FA*L[IE]/(-EXP(
                        II*LN(ABS(L[I]-L[IE]))))
                    END;
                END;
            END;
        END;
    END;
    A[IE,1]:=SUM+A[IE,1];
END;

PROCEDURE SUB1;
BEGIN
  I:=1;
  WHILE I<=K DO
    BEGIN
      IF I<>IE THEN
        BEGIN
          A[I,NI[I]]:=A[I,NI[I]]*L[IE]/(L[IE]-L[I]);
          IF NI[I]<>1 THEN
            BEGIN
              J:=1;
              NNI:=NI[I]-J;
              WHILE J<=(NI[I]-1) DO
                BEGIN
                  A[I,NNI]:=(L[IE]*A[I,NNI]-NNI*A[I,NNI+1])/
                  (L[IE]-L[I]);
                  J:=J+1;
                  NNI:=NI[I]-J;
                END;
            END;
        END;
      I:=I+1;
    END;
END;

PROCEDURE ONEATATIME;
BEGIN
  IF NINIT[IE]=NI[IE] THEN
    IE:=IE+1
  ELSE
    BEGIN
      NI[IE]:=NI[IE]+1;
      J:=0;
      REPEAT
        NN:=NI[IE]-J;
        A[IE,NN]:=L[IE]*A[IE,NN-1]/(NN-1);
        J:=J+1;
```

```
      UNTIL NN=2;
      SUMMATION;
      SUB1;
    END;
END;

PROCEDURE SUB2;
BEGIN
  RELI:=0.0;
  IF T[IT]=0 THEN
    RELI:=1.0
  ELSE
    BEGIN
      FOR I:=1 TO K DO
        BEGIN
          SUM:=0.0;
          NKK:=NI[I];
          FOR J:=1 TO NKK DO
            SUM:=SUM+A[I,J]*EXP((J-1)*LN(T[IT]));
          PRO[I]:=SUM*EXP(-L[I]*T[IT]);
        END;
        FOR I:=1 TO K DO
          RELI:=RELI+PRO[I];
    END;
    IF RELI>1.0 THEN
      REL[IT]:=1.0
    ELSE
      REL[IT]:=RELI;
END;

PROCEDURE PREPFORLOOP;
BEGIN
  FOR I:=1 TO MAX DO
    NINIT[I]:=-1;
  KKK:=1;
  I:=1;
  IF L[2]<0 THEN
    BEGIN
      NINIT[1]:=1;
      K:=1;
    END;
  WHILE L[I+1]>0 DO
    BEGIN
      IF L[I]<L[I+1] THEN
        BEGIN
          NINIT[KKK]:=1;
          K:=KKK;
          KKK:=KKK+1;
          I:=I+1;
          IF L[I+1]<0 THEN
            NINIT[KKK]:=1;
            K:=KKK;
```

```
   IF (X=-1) AND (Y=8) THEN PRINTVAR:='4';
   IF (X=-1) AND (Y=6) THEN PRINTVAR:='3';
   IF (X=-1) AND (Y=4) THEN PRINTVAR:='2';
   IF (X=-1) AND (Y=2) THEN PRINTVAR:='1';
END;

PROCEDURE GRAPHICS;
BEGIN
  COUNT:=1;
  FOR Y:=20 DOWNTO 1 DO
    BEGIN
      FOR X:=-3 TO 76 DO
        BEGIN
          IF COUNT<=TMAX THEN
            BEGIN
              VAL:=ROUND(20*REL[COUNT]);
              IF (VAL=Y) AND (X=T[COUNT]) THEN
                BEGIN
                  PRINTVAR:=ASTERISK;
                  COUNT:=COUNT+1;
                END
              ELSE
                PRINTVAR:=BLANK;
            END
          ELSE
            PRINTVAR:=BLANK;
          YAXIS;
          WRITE(PRINTVAR);
        END;
  END;
  Y:=0;
  FOR X:=-3 TO 76 DO
    BEGIN
      IF (X MOD 5 =0) THEN PRINTVAR:=LINE;
      IF COUNT<=TMAX THEN
        BEGIN
          VAL:=ROUND(20*REL[COUNT]);
          IF (VAL=Y) AND (X=T[COUNT]) THEN
            BEGIN
              PRINTVAR:=ASTERISK;
              COUNT:=COUNT+1;
            END
          ELSE
            IF (X MOD 5 <> 0) THEN
              PRINTVAR:=DASH;
        END
      ELSE
        IF (X MOD 5 <> 0) THEN
          PRINTVAR:=DASH;
      IF (X=-3) THEN PRINTVAR:='0';
      IF (X=-2) THEN PRINTVAR:='.';
      IF (X=-1) THEN PRINTVAR:='0';
```

```
        END
      ELSE
        BEGIN
          II:=I;
          COUNT:=1;
          WHILE L[II]=L[II+1] DO
            BEGIN
              COUNT:=COUNT+1;
              NINIT[KKK]:=COUNT;
              FOR IL:=II TO 9 DO
                L[IL]:=L[IL+1];
              L[MAX]:=-1;
              K:=KKK;
            END;
          KKK:=KKK+1;
          I:=II+1;
          IF (((COUNT>1) AND (L[I+1]<0) AND (L[I]>0))) THEN
            BEGIN
              NINIT[KKK]:=1;
              K:=KKK;
            END;
        END;
    END;
END;

PROCEDURE LOOP;
BEGIN
FOR IT:=1 TO TMAX DO
  BEGIN
    INITIALIZE;
    REPEAT
      ONEATATIME;
    UNTIL IE=K+1;
    SUB2;
    REL[IT]:=REL[IT]*P[BR];
    RELTOT[BR,IT]:=REL[IT];
  END;
END;

PROCEDURE YAXIS;
BEGIN
  IF (X=0) THEN PRINTVAR:=LINE;
  IF ((X=0) AND (Y=20)) AND (T[1]=0) THEN PRINTVAR:=ASTERISK;
  IF (X=-3) AND (Y MOD 2 = 0) THEN PRINTVAR:='0';
  IF (X=-3) AND (Y=20) THEN PRINTVAR:='1';
  IF (X=-2) AND (Y MOD 2=0) THEN PRINTVAR:='.';
  IF (X=-1) AND ((Y=20) OR (Y=0)) THEN PRINTVAR:='0';
  IF (X=-1) AND (Y=18) THEN PRINTVAR:='9';
  IF (X=-1) AND (Y=16) THEN PRINTVAR:='8';
  IF (X=-1) AND (Y=14) THEN PRINTVAR:='7';
  IF (X=-1) AND (Y=12) THEN PRINTVAR:='6';
  IF (X=-1) AND (Y=10) THEN PRINTVAR:='5';
```

```
      WRITE(PRINTVAR);
    END;
  WRITE('    0    5    10    15    20    25    30    35    40    45
  50    55    60');
  WRITELN('     TIME -->');
END;

PROCEDURE DECIDE;
BEGIN
  WRITELN;WRITELN;
  WRITELN('DO YOU WANT TO CONTINUE WITH MIXED CONVOLUTIONS?
  TYPE "U" :');
  WRITELN('OR DO YOU WANT TO COMPUTE CONVOLUTIONS?
  TYPE "XP1" :');
END;

BEGIN
  INPUT;
  OUTPUT1;
  GETDATA;
  OUTPUT2;
  COMPPROB;
  COMPLAMB;
  FOR I:=1 TO MAX DO
    L[I]:=-1;
  FOR BR:=1 TO BRANCHES DO
    BEGIN
      J:=1;
      WHILE LAMB[BR,J]>=0 DO
        BEGIN
          IF LAMB[BR,1]>0 THEN
            BEGIN
              L[J]:=LAMB[BR,J];
              J:=J+1;
            END
          ELSE
            BEGIN
              L[J]:=LAMB[BR,J+1];
              J:=J+1;
            END;
        END;
      SORT;
      PREPFORLOOP;
      LOOP;
    END;
    FOR I:=1 TO TMAX DO
      BEGIN
        SUMREL[I]:=0;
        FOR BR:=1 TO BRANCHES DO
          SUMREL[I]:=RELTOT[BR,I]+SUMREL[I];
        WRITELN('RELIABILITY ( T > ',T[I]:3,' ) = ',
        SUMREL[I]:8:6);
```

```
      END;
    WRITELN;
  ATTENTION;
  WRITELN('DO YOU ALSO WANT A GRAPHICAL PRESENTATION
  OF THE DATA?');
  WRITE('HIT "1" FOR YES OR "0" FOR NO :  ');
  READLN(CHGR);
  WRITELN;
  WRITELN;
  IF CHGR=1 THEN
    GRAPHICS;
  DECIDE;
END.
```

## LIST OF REFERENCES

1. Gursel, S., Some Computer Algorithms to Implement A Reliability Shorthand, M.S. Thesis, Naval Postgraduate School, Monterey, CA 93943, 1982.

# BIBLIOGRAPHY

Barlow, R.E. and Proschan, F., Statistical Theory of Reliability and Live Testing, Silver Springs, 1981.

Esary, J.D., Course Notes and Problem Sets for OA 4302, Naval Postgraduate School, Monterey, CA 93943, 1982.

Peters, H.-E., Implementation of a Reliability Shorthand on the TI-59 Handheld Calculator, M.S. Thesis, Naval Postgraduate School, Monterey, CA 93943, 1982.

Repicky, John J., Jr., An Introduction to a Reliability Shorthand, M.S. Thesis, Naval Postgraduate School, Monterey, CA 93943, 1982.

## INITIAL DISTRIBUTION LIST

| | | No. Copies |
|---|---|---|
| 1. | Defense Technical Information Center<br>Cameron Station<br>Alexandria, Virginia 22314 | 2 |
| 2. | Library, Code 0142<br>Naval Postgraduate School<br>Monterey, California 93943 | 2 |
| 3. | Department Chairman, Code 55<br>Naval Postgraduate School<br>Monterey, California 93943 | 1 |
| 4. | Professor J.D. Esary, Code 55Ey<br>Department of Operations Research<br>Naval Postgraduate School<br>Monterey, California 93943 | 1 |
| 5. | Professor A.F. Andrus, Code 55Ad<br>Department of Operations Research<br>Naval Postgraduate School<br>Monterey, California 93943 | 1 |
| 6. | Captain Eckhard Bartens<br>Schatzkammer 27<br>D - 3220 Alfeld<br>West Germany | 1 |
| 7. | Major Hans-Eberhard Peters<br>Hemmetworth 2 Beckedorf<br>D - 3102 Hermannsburg<br>West Germany | 1 |

205358

Thesis
B24229 Bartens
c.1 Implementation of a reliability shorthand on the Apple II plus microcomputer.